# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/864,973 | 05/24/2001 | Seung Jae Chung | 8729-206 (IB200010-238) | 3440 |

7590 03/29/2004

Frank Chau
1900 Hempstead Turnpike, Suite 501
East Meadow, NY 11554

| EXAMINER | |
|---|---|
| GOLE, AMOL V | |
| **ART UNIT** | **PAPER NUMBER** |
| 2183 | 4 |

DATE MAILED: 03/29/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| *Office Action Summary* | **Application No.** 09/864,973 | **Applicant(s)** CHUNG ET AL. | |
|---|---|---|---|
| | **Examiner** Amol V. Gole | **Art Unit** 2183 | |

***-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --***

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on _5/24/01, 8/20/01, 7/23/02_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) _1-24_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-24_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _24 May 2001_ is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _2,3_.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

## DETAILED ACTION

1. Claims 1-24 have been examined.

### *Papers Submitted*

2. It is hereby acknowledged that the following papers have been received and placed of record in the file:

#2: IDS (8/20/01)

#3: IDS (7/23/02)

### *Information Disclosure Statement*

3. The information disclosure statement filed 7/23/02 fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each U.S. and foreign patent; each publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. It has been placed in the application file, but the information referred to therein has not been considered. The foreign patent document JP62063334 is listed but not received. Also a document JP 62063344 was received but not listed in the IDS; it also will not be considered.

### *Drawings*

4. The drawings are objected to because in fig. 1 lines 155 and program address line that is an input to the loop buffer 124 are shown to thinner as compared to the other data lines. This inconsistency may lead to confusion with say a control line.

A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

### *Specification*

5. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The

disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

6. The abstract contains legal phraseology and should be changed appropriately.

7. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required:

In claim 7 the limitation of storing the first instruction within the loop in any of the m registers of the loop buffer. However the specification calls for storing the first instruction in the specific register that is addressed using the LSBs of the program counter storing the address of the that first instruction (pg. 12, lines 7-12) and not any of the m registers. Please either modify the claim or show the antecedent basis in the specification.

8. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: LOOP INSTRUCTION PROCESSING IN RESPONSE TO A COPROCESSOR-TYPE LOOP OPERATION USING A LOOP BUFFER IN A DATA PROCESSING DEVICE.

### *Claim Objections*

9. Claim 3 is objected to because of the following informalities: It is unclear whether the limitation of "said step of decoding" is referring to the CPU decoding step (pg. 15, line 4) of the coprocessor decoding step (pg. 15, line 6).

Appropriate correction is required.

10. Claim 20 is objected to because of the following informalities: pg. 19, lines 1: the word multiplexer is misspelled as "multiplexor".

Appropriate correction is required.

11. Claim 24 is objected to because of the following informalities: On pg. 20, line 1, replace "retrieve" with "retrieved" for grammatical purposes.

Appropriate correction is required.

### *Claim Rejections - 35 USC § 112*

12. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

13. Claim **20** rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

14. Claim 20 recites the limitation "the loop buffer flags" in pg. 19, lines 1-2. There is insufficient antecedent basis for this limitation in the claim.

It seems that claim 20 should be dependant on any of claim 16-18 as they disclose the limitation of "loop buffer flags". For purposes of the following art rejection, claim 20 will be made dependant on claim 16. Please make the appropriate correction in any case.

### *Claim Rejections - 35 USC § 103*

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

*(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

16. Claims **1-7, 14, 15, 19, 21, 23, and 24** are rejected under 35 U.S.C. 103(a) as being unpatentable over Kim et al. (US006532530B1) in view of Kiuchi et al. (US005579493A).

**17. In regard to claim 1:**

18. Kim et al. teaches a method of processing loop instructions (col. 12, lines 49-58) using a data processing device having a central processing unit (CPU) (fig. 3, host processor 30) and a coprocessor (32), wherein the CPU fetches and decodes instructions retrieved from program memory and determines whether the instructions are CPU-type or coprocessor-type (col. 6, lines 33-38), comprising the steps of:

decoding the coprocessor-type instructions by the coprocessor (col. 6, lines 46-57).

19. However Kim et al. do not teach the steps of if a loop operation is decoded by the coprocessor, retrieving from the program memory the instructions within the loop, storing the retrieved instructions within the loop in a loop buffer, and then inhibiting instruction fetch from the program memory while instructions within the loop are executed in a subsequent iteration of the loop.

20. Kiuchi et al. teach a method of processing loop instructions using a data processor (fig. 1, 100) with repeat control circuitry (107) and an instruction buffer (108). The processor on fetching and decoding a loop instruction, initiates storing the instructions in the loop into the instruction buffer 108 so that on subsequent execution of the loop, the instructions are fetched from the instruction buffer 108 and not the program memory 101 in order to reduce power consumption (col. 3, lines 23-27; col. 6, lines 8-14, 46-64). Also on subsequent iterations of the loop, the fetching for the program memory is disabled (col. 6, lines 50-57)

21. One of ordinary skill in the art at the time of the invention would have recognized that as an efficient looping scheme is important to the DSP coprocessor (as indicated in Kim: col. 12, lines 49-50) and that as one of the objectives of the CPU-coprocessor system is low-power (Kim: col. 3, lines 53-57) that it would be advantageous to have a loop buffer (Kiuchi: instruction buffer 108) for executing loops. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Kim et al. system by adding the loop buffer and its required control logic and have the DSP coprocessor handle the loop buffer scheme; the coprocessor on decoding a loop instruction would, just like Kiuchi et al.'s system, store the instructions in the loop into a loop buffer (Kiuchi: Instruction buffer 108) from the program memory and also inhibit fetching from the program memory on subsequent iterations of the loop (Kiuchi: col. 6, lines 50-57).

22. It would have been obvious to do so because it overall leads to a system consuming even less power while showing better loop execution performance.

**23. In regard to claim 2:**

24. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 1, further including the step of accessing the instructions within the loop from the loop buffer in a subsequent iteration of the loop (Kiuchi: col. 6, lines 53-55).

**25. In regard to claim 3:**

26. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 1, wherein said step of decoding further includes determining a backward branch distance for use by the CPU to control branching to and from the loop (Kiuchi: col. 7, 1-8 show that the number of steps in the loop are determined on decoding the instruction. This is inherently the backward branch distance because the number of steps in the loop indicate the last instruction in the loop from which the loop backward branches to the start of the loop as per normal execution of a loop).

**27. In regard to claim 4:**

28. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 1 further including the steps of:

determining from the loop instruction a number of iterations of the loop operation (Kiuchi: operand signal 117c; col. 7, lines 1-8);

decrementing by the coprocessor the number of iterations upon completion of each loop (Kiuchi: col. 7, lines 24-26, 33-40); and

signaling to the CPU the completion of the loop operation when reaching the end of the number of iterations (Kiuchi: col. 10, lines 20-36).

**29. In regard to claim 5:**

30. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 1, wherein said storing step includes storing `n` loop instructions in `m` registers of the loop buffer and addressing the `m` registers by $\log_2 m$ least significant bits (LSBs) of a program counter which is also used for addressing the program memory (Kiuchi: col. 7, lines 45-59), wherein n or m is any natural number and n is less than or equal to m (Inherently the number of loop instructions 'n' to be stored in loop buffer has to be less than or equal to the number of registers of the loop buffer 'm' because you can not store more than 'm' loop instructions in the loop buffer).

**31. In regard to claim 6:**

32. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 5, further including the steps of accessing the instructions stored in the loop buffer through a multiplexer (Kiuchi: fig. 1, 102) and controlling the multiplexer output by the $\log_2 m$ LSBs of the program counter (Kiuchi: the $\log_2 m$ LSBs of the program counter (118a, col. 7, lines 50-54) are used to select the loop instruction from the instruction buffer and this loop instruction is sent to the multiplexer 102 to be outputted [col. 8, lines 10-20]. Hence the LSBs of the program counter control the multiplexer output).

**33. In regard to claim 7:**

34. The combination of Kim et al. in view of Kiuchi et al. teaches the method of claim 5, wherein a first instruction within the loop is stored in any of the m registers addressed by the LSBs of the program counter (col. 7, lines 60+ and col. 8, lines 1-9 teach that the LSBs of the program counter (118a, col. 7, lines 50-54) address the register into which the instruction within the loop, which inherently includes the first instruction, is stored. Depending on the LSBs, this could be any of the m registers).

**35. In regard to claim 14:**

36. Kim et al. teach a data processing device comprising:

a central processing unit (CPU) (fig. 3, host processor 30) for fetching instructions from a program memory (34), decoding the instructions (col. 6, lines 33-38) and sending a signal (CCLK) to a coprocessor if a coprocessor type instruction is decoded (Active signal B indicates that signal A is a coprocessor operation [col. 6, lines 47-57]);

a coprocessor for decoding the coprocessor-type instructions (signal A) upon receipt of the signal (CCLK) (col. 6, lines 55-57).

37. However Kim et al. do not teach a loop buffer for receiving from the program memory instructions within a loop and storing the instructions within the loop when the coprocessor decodes a loop operation from the coprocessor-type instructions, wherein the instructions within the loop are retrieved from the loop buffer for execution in a subsequent iteration of the loop.

38. Kiuchi et al. teach a system using a data processor (fig. 1, 100) with repeat control circuitry (107) and an instruction buffer (108) for processing loop instructions. The processor on fetching and decoding a loop instruction from the program memory 101, initiates storing the instructions in the loop into the instruction buffer 108 so that on subsequent iteration of the loop, the instructions are fetched from the instruction buffer 108 and not the program memory 101 in order to reduce power consumption (fig. 1; col. 3, lines 23-27; col. 6, lines 8-14, 46-64).

39. One of ordinary skill in the art at the time of the invention would have recognized that as an efficient looping scheme is important to the DSP coprocessor (as indicated in Kim: col. 12, lines 49-50) and that as one of the objectives of the CPU-coprocessor system is low-power (Kim: col. 3, lines 53-57) that it would be advantageous to have a loop buffer (Kiuchi: instruction buffer 108) for executing loops. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Kim et al. system by adding the loop buffer and its required control logic and have the DSP coprocessor handle the loop buffer scheme; the coprocessor on decoding a loop instruction would, just like Kiuchi et al.'s system, store the instructions in the loop into a loop buffer (Kiuchi: Instruction buffer 108) from the program memory and on subsequent iteration of the loop, would fetch the instructions from the loop buffer and not the program memory.

40. It would have been obvious to do so because it overall leads to a system consuming even less power while showing better loop execution performance.

**41. In regard to claim 15:**

42. The combination of Kim et al. in view of Kiuchi et al. teaches the device of claim 14, wherein a disable signal is sent to the program memory for inhibiting access of the program memory while the instructions within the loop are retrieved from the loop buffer (Kiuchi: signal 122; col. 6, lines 50-57).

**43. In regard to claim 19:**

44. The combination of Kim et al. in view of Kiuchi et al. teaches the device of claim 14, wherein the loop buffer includes `m` registers and the registers are addressed by $\log_2 m$ LSBs of a program counter used for addressing the program memory (Kiuchi: col. 7, lines 45-59).

**45. In regard to claim 21:**

46. The combination of Kim et al. in view of Kiuchi et al. teaches the device of claim 14, wherein the coprocessor decodes from a loop instruction a loop block size (Kiuchi: 117b; col. 7, lines 1-6) and a number of iterations of looping (Kiuchi: 117c; col. 7, lines 1-8), and calculates a backward branch distance for use by the CPU to control branching to and from the loop. (Kiuchi: col. 7, 1-8 show that the number of steps in the loop are determined on decoding the instruction. The number of steps in the loop is inherently the backward branch distance because the last instruction in the loop from which the loop backward branches to the start of the loop, as per normal execution of a loop, is at a distance equal to the number of steps in the loop)

**47. In regard to claim 23:**

48. The combination of Kim et al. in view of Kiuchi et al device of claim 14, wherein the instructions stored in the loop buffer comprise coprocessor and CPU type instructions (Kim et al. teaches the use of a SOC (System-on-Chip) approach wherein the CPU (microprocessor 30) is integrated with the DSP coprocessor (32) which can execute both normal microprocessor based instructions and DSP coprocessor instructions [col. 2, lines 25-40]. Hence, as the program memory contains both CPU type and coprocessor type instructions, a loop or any stream of instructions can contain both CPU type and coprocessor type instructions. Also as shown in fig. 3, the coprocessor 32 also can send data back to the host processor 30 via bus E. Therefore combination of Kim et al. and Kiuchi et al. teaches that the loop buffer comprises of both the coprocessor and CPU type instructions).

**49. In regard to claim 24:**

50. data processing device comprising:

a central processing unit (CPU) (fig. 3, host processor 30) for fetching instructions from a program memory (34), decoding the instructions (col. 6, lines 33-38) and sending a signal (CCLK) to a coprocessor if a coprocessor type instruction is decoded (Active signal B indicates that signal A is a coprocessor operation [col. 6, lines 47-57]);

a coprocessor for decoding the coprocessor-type instructions upon receipt of the signal (CCLK) (col. 6, lines 55-57).

51. However Kim et al. do not teach a loop buffer for receiving from the program memory instructions within a loop and storing the instructions within the loop when the coprocessor decodes a loop operation from the coprocessor-type instructions, wherein the instructions within the loop are retrieved from the loop buffer for execution in a subsequent iteration of the loop, wherein a disable signal is sent to the program memory for inhibiting access of the program memory while the instructions within the loop are retrieved from the loop buffer.

52. Kiuchi et al. teach a system using a data processor (fig. 1, 100) with repeat control circuitry (107) and an instruction buffer (108) for processing loop instructions. The processor on fetching and decoding a loop instruction from the program memory 101, initiates storing the instructions in the loop into the instruction buffer 108 so that on subsequent iteration of the loop, the instructions are fetched from the instruction buffer 108 and not the program memory 101 in order to reduce power consumption (fig. 1; col. 3, lines 23-27; col. 6, lines 8-14, 46-64). Also on subsequent iterations of the loop, the fetching for the program memory is disabled (col. 6, lines 50-57).

53. One of ordinary skill in the art at the time of the invention would have recognized that as an efficient looping scheme is important to the DSP coprocessor (as indicated in Kim: col. 12, lines 49-50) and that as one of the objectives of the CPU-coprocessor system is low-power (Kim: col. 3, lines 53-57) that it would be advantageous to have a loop buffer (Kiuchi: instruction buffer 108) for executing loops. Therefore it would have

been obvious to one of ordinary skill in the art at the time of the invention to have modified the Kim et al. system by adding the loop buffer and its required control logic and have the DSP coprocessor handle the loop buffer scheme; the coprocessor on decoding a loop instruction would, just like Kiuchi et al.'s system, store the instructions in the loop into a loop buffer (Kiuchi: Instruction buffer 108) from the program memory and on subsequent iteration of the loop, would fetch the instructions from the loop buffer and not the program memory and also inhibit fetching from the program memory on subsequent iterations of the loop (Kiuchi: col. 6, lines 50-57).

54. It would have been obvious to do so because it overall leads to a system consuming even less power while showing better loop execution performance.

55. Claims **8-13, 16-18, and 20** are rejected under 35 U.S.C. 103(a) as being unpatentable over Kim et al. in view of Kiuchi et al. as applied to claims 1-7 above, and further in view of Moyer et al. (US005920890A).

**56. In regard to claim 8:**

57. The combination of Kim et al. in view of Kiuchi et al. teaches the select signal 114 used to indicate the presence or absence of active loop instructions in the loop buffer to the selector 102 (Kiuchi: fig. 1) but does not teach the method of claim 1, further including the steps of signaling the presence or absence of an active loop instruction by a loop buffer flag in each of the `m` registers in the loop buffer, the presence of an active instruction in a register is indicated by a preassigned signal in the loop buffer flag.

58. Moyer et al. teach a loop cache (fig. 1, 26) where each of the registers 52 in the loop cache have a valid bit 54 indicating the presence or absence of a new active loop instruction (col. 3, lines 45-49). Every time a new entry is loaded into the register array 53, the corresponding valid bit is set to '1'. This valid bit is used to generate the loop cache hit signal that is used to select the input that is to be outputted by the multiplexer 28 (col. 3, lines 15-23) when selected by the LCACHE index (col. 3, lines 33-35).

59. One or ordinary skill in the art would have recognized that by using a valid bit for every register in the loop buffer (register array 53) like Moyer (col. 3, lines 45-49), it would allow one to replace individual entries in the loop buffer and therefore allow one to have more than one loop in the buffer at the same time instead of the case where you

have only one indicator 114 as in Kiuchi et al. for the entire buffer. Also, *In re Harza*, 274 F.2d 669, 671, 124 USPQ 378, 380 (CCPA 1960) addresses this, viz., duplicating part for a multiple effect. By duplicating the selection signal 114 for each of the loop buffer register entries, one would get fine grain control over the selection. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to add a loop buffer flag (Moyer: valid bit 54) for each register of the loop buffer (Kiuchi: instruction buffer 108) to indicate the presence or absence of an active loop instruction wherein the presence of an active instruction is indicated by setting it to '1'.

**60. In regard to claim 9:**

61. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the method of claim 8, further including the step of accessing each flag in the loop buffer by $\log_2 m$ least significant bits of a program counter used for addressing the program memory (Moyer teaches that the LCACHE index is used to access each flag (valid bit) and further fig. 2 and col. 4, lines 6-12 teach that the LCACHE index is the $\log_2 m$ LSBs of the program counter).

**62. In regard to claim 10:**

63. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the method of claim 8, further including the step of multiplexing an instruction from the loop buffer and the program memory, the multiplexing is dependent upon a presence of an active instruction signal from a loop buffer flag (Moyer teaches that the

multiplexing is dependant on the loop cache hit signal which is generated using the flag (valid bit) col. 3, lines 15-22).

**64. In regard to claim 11:**

65. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the method of claim 8, wherein said step of inhibiting instruction fetch from the program memory includes sending an inhibit signal to the program memory when the preassigned signal in the loop buffer flag is read and indicates the presence of an active loop instruction (Kuichi teaches in col. 6, lines 50-57 that on subsequent iterations of the loop instructions, the inhibit signal (122) is sent to the program memory 101 to disable access to the program memory so that the instructions can be read from the loop buffer. Moyer teaches in col. 3, lines 45-49 that when a new loop instruction is loaded in the loop buffer, the loop buffer flag (valid bit) is set to '1'. Therefore, when a loop buffer register is accessed and if loop buffer flag is set to one that means that this is the subsequent iteration of that instruction. Hence, inherently the combination teaches that the inhibit signal is sent to the program memory when then loop buffer flag is read and indicates the presence of an active instruction i.e. it is set to '1').

**66. In regard to claim 12:**

67. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the method of claim 11, wherein the preassigned signal in each of said loop buffers is selectively alterable by the CPU independent of the presence or absence of an active instruction in corresponding registers (Moyer: col. 4, lines 24-29).

**68. In regard to claim 13:**

69. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the method of claim 8, further including the step of clearing the loop buffer flag when the loop operation is completed (Moyer: fig. 3 shows the step 78 wherein the loop buffer flag is cleared (invalidate loop cache) when the loop operation is completed (GTAG is not hit i.e. next instruction address is outside the loop meaning the loop has completed)).

**70. In regard to claim 16:**

71. The combination of Kim et al. in view of Kiuchi et al. teaches the device of claim 14, wherein the loop buffer includes `m` registers (fig. 3, shows instruction buffer 108 with 'm' registers 301).

72. However it does not teach that each register has a corresponding loop buffer flag for indicating whether the corresponding register is filled with an instruction.

73. Moyer et al. teach a loop cache (fig. 1, 26) where each of the registers 52 in the loop cache have a valid bit 54 indicating the presence or absence of a new active loop

instruction (col. 3, lines 45-49). Every time a new entry is loaded into the register array 53, the corresponding valid bit is set to '1'. This valid bit is used to generate the loop cache hit signal that is used to select the input that is to be outputted by the multiplexer 28 (col. 3, lines 15-23) when selected by the LCACHE index (col. 3, lines 33-35).

74. One or ordinary skill in the art would have recognized that by using a valid bit for every register in the loop buffer (register array 53) like Moyer (col. 3, lines 45-49), it would allow one to replace individual entries in the loop buffer and therefore allow one to have more than one loop in the buffer at the same time instead of the case where you have only one indicator 114 as in Kiuchi et al. for the entire buffer. Also, *In re Harza*, 274 F.2d 669, 671, 124 USPQ 378, 380 (CCPA 1960) addresses this, viz., duplicating part for a multiple effect. By duplicating the selection signal 114 for each of the loop buffer register entries, one would get fine grain control over the selection. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to add a loop buffer flag (Moyer: valid bit 54) for each register of the loop buffer (Kiuchi: instruction buffer 108) to indicate the presence or absence of an active loop instruction wherein the presence of an active instruction is indicated by setting it to '1'.

**75. In regard to claim 17:**

76. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the device of claim 16, wherein the loop buffer flags are accessed by $\log_2 m$ least significant bits of a program counter used for addressing the program memory (Moyer teaches that the LCACHE index is used to access each flag (valid bit) and

further fig. 2 and col. 4, lines 6-12 teach that the LCACHE index is the $\log_2 m$ LSBs of the program counter).

**77. In regard to claim 18:**

78. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the device of claim 16, wherein a program memory inhibit signal is generated based on a signal read from the loop buffer flag (Kuichi teaches in col. 6, lines 50-57 that on subsequent iterations of the loop instructions, the inhibit signal (122) is sent to the program memory 101 to disable access to the program memory so that the instructions can be read from the loop buffer. Moyer teaches in col. 3, lines 45-49 that when a new loop instruction is loaded in the loop buffer, the loop buffer flag (valid bit) is set to '1'. Therefore, when a loop buffer register is accessed and if loop buffer flag is set to one that means that this is the subsequent iteration of that instruction. Hence, inherently the combination teaches that the inhibit signal is sent to the program memory based on the loop buffer flag read which indicates the presence of an active instruction when it is set to '1').

**79. In regard to claim 20:**

80. The combination of Kim et al. in view of Kiuchi et al. in further view of Moyer et al. teaches the device of claim ***16***, further including a multiplexer for multiplexing between the instructions retrieved from the program memory and the instructions retrieved from the loop buffer, the multiplexer being controlled by signals read from the loop buffer

flags (Moyer teaches that the multiplexer 28 for multiplexing between the instructions from main memory 24 and loop cache 26 is controlled by the loop cache hit signal which is generated using the loop buffer flag (valid bit) [fig. 1; col. 3, lines 15-22]).

81. Claim **22** is rejected under 35 U.S.C. 103(a) as being unpatentable over Kim et al. in view of Kiuchi et al. as applied to claim 21 above, and further in view of Hsu et al. (US005854934A).

**82. In regard to claim 22:**

83. The combination of Kim et al. in view of Kiuchi et al. does not teach the device of claim 21, wherein the backward branch distance is the loop block size minus one.

84. Hsu et al. teach a branch delay slot which is a slot following the branch instruction in which a useful instruction is filled by the compiler. This allows the branch condition to be resolved and the target to be determined one cycle in advance so that there is no loss in performance due to the control hazard created by the branch.

85. One of ordinary skill in the art would have recognized that by placing a branch delay slot after the backward branching instruction at the end of the loop, one could gain some performance. However now the backward branch distance would equal to the number of steps in the loop minus one because the branch instruction is now at one location before the end of the loop. Therefore it would have been obvious to one of ordinary skill in the art to calculate the backward branch distance as the loop block size (number of steps in the loop) minus one.

86. One would have been motivated to do so because by using the branch delay slot technique one would gain performance but would in turn calculate the backward branch distance as the loop block size minus one.

## *Conclusion*

87. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty, which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections. See 37 CFR § 1.111.

a. Fleck et al. (US006085315A) teach a loop cache addressed by the program counter.

b. Emma et al. (US004991080) teaches a branch stream coprocessor.

c. Getzlaff et al. (US005634047A) teaches a method of processing loop end conditions in a second processor.

d. Osovets (US006125440A) teaches a method of reducing power consumption by storing instructions of a loop in a shift register and on executing a backward branch instruction, executing the instructions in the shift register instead of the RAM.

e. Fernando et al. (US006269440B1) teaches a loop buffer that on execution of a VDO loop instruction, sets the instruction count and the number of iteration in registers and can handle multiple issue packets.

f. Ganapathy et al. (US006598155B1) teach a loop buffering technique for DSP instructions.

g. Le Cornec (US006427203B1) teaches an instruction format advantageous to looping in DSP applications.

h. Agarwal et al. (US 20020156994A1) teaches benefits of using a DSP coprocessor.

88. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Amol V. Gole whose telephone number is 703-305-8888. The examiner can normally be reached on 9:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 703-305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AVG
amol.gole@uspto.gov

Eddie C

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100